EXHIBIT A

System.Firewall.P licy.ApplicationRule

```
namespace System.Firewall.Policy
{
  public class ApplicationRule : PolicyRule
  {
    public ApplicationRule();
    public ApplicationRule(ApplicationConditon condition, ApplicationAction act);

    public ApplicationCondition ApplicationCondition { get { } set { } }
    public ApplicationAction Action { get { } set { } }
  }
}
```

ApplicationRule is the rule that is enforced by the application layer enforcement although it may also dynamically instantiate rules at other layers e.g. an IPSec rule the transport layer to secure all traffic by a certain application or user.

System.Firewall.Policy.TransportRule

```
namespace System.Firewall.Policy
{
    public class TransportRule : PolicyRule
    {
        public TransportRule();
        public TransportRule(TransportConditon condition, FilterAction act);
        public TransportCondition TransportCondition { get { } set { } }
        public FilterAction Action { get { } set { } }
    }
}
```

TransportRule models the traditional firewall rule that mainly filters on the standard 5-tuple.

System.Firewall.Policy.IKERule

```
namespace System.Firewall.Policy
{
    public class IKERule : PolicyRule
    {
        public IKERule();
        public IKERule(IPAddressValue src, IPAddressValue dst, IKEAction act);
        public IPAddressValue SourceAddress { get { } set { } }
        public IPAddressValue DestinationAddress { get { } set { } }
        public IKEAction Action { get { } set { } };
    }
}
```

There are three different rules for specifying IPSec related policies: IPSecRule, KeyingModuleRule and IKERule. IPSecRule is added at the transport layer where matching traffic triggers the IPSec callout. The IPSec callout set a security context in the packet so that the IPSec module will be invoked to search for existing SAs to secure the traffic. If none is found, KeyingModuleRule will be matched to find the right keying module to perform key negotiation. Depending on the keying module selected, the corresponding IKERule or MamieRule will be matched to find the appropriate configure settings for performing the key exchange. Then IPSecRule will again be matched to set up the proper IPSec SA that will be used for actually securing the traffic e.g. AH or ESP.

IKERule specifies the parameters for carrying out IKE key negotiation protocol. **IKERule** can only take local address and remote address as its condition. The action for **IKERule** is an **IKEAction**.

System.Firewall.P licy.IPSecRule

```
namespace System.Firewall.Policy
{
      public class IPSecRule : PolicyRule
      {
            public IPSecRule();
            public IPSecRule(IPAddressValue srcAddr, IPAddressValue dstAddr,
             ByteValue protocol, UInt16Value srcPort, UInt16Value dstPort,
             IPSecAction action);
            public IPAddressValue SourceAddress { get { } set { } }
            public IPAddressValue DestinationAddress { get { } set { } }
            public ByteValue Protocol { get { } set { } }
            public UInt16Value SourcePort { get { } set { } }
            public UInt16Value DestinationPort { get { } set { } }
            public IPSecAction Action { get { } set { } };
      }
}
```

Conceptually **IPSecRule** plays two distinct roles, one is to trigger the IPSec callout when the associating condition is matched, and another to indicate the configure parameters for securing the matching traffic. So it specifies both what packets need to be secured and also how they will be secured. Different 5 tuples can have different IPSec parameters. Although at the transport layer, the firewall platform can match more fields than the standard 5-tuple e.g. TCP flags, for the purpose of carrying out IPSec, 5-tuple is sufficient. So **IPSecRule** only lists the standard 5 tuple, i.e. source address, destination address, protocol, source port and destination port, as its condition fields. Other fields can be added if there are practical cases requires traffics matching the same 5-tuple to be secured differently based on those fields.

System.Firewall.Policy.KeyingM duleRule

```
namespace System.Firewall.Policy
{
    public class KeyingModuleRule : PolicyRule
    {
        public KeyingModuleRule();
        public KeyingModuleRule(IPAddressValue srcAddr, IPAddressValue
          dstAddr, ByteValue protocol, UInt16Value srcPort, UInt16Value
          dstPort, KeyingModuleAction action);
        public IPAddressValue SourceAddress { get { } set { } }
        public IPAddressValue DestinationAddress { get { } set { } }
        public ByteValue Protocol { get { } set { } }
        public UInt16Value SourcePort { get { } set { } }
        public UInt16Value DestinationPort { get { } set { } }
        public KeyingModuleAction Action { get { } set { } };
    }
}
```

KeyingModuleRule selects which key negotiation module to use when there is no existing secure channel (association) to the remote peer, which could be a host or a user. **KeyingModuleRule** also take the standard 5-tuple as its condition. In case where more than one module is available, for example Mamie for user authentication in addition to traditional IKE, **KeyingModuleAction** lists them in the order that they will be tried either concurrently or sequentially until one of them succeeds or all fails.